AF

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: David Yach

Art Unit: 2153

Application No.: 09/728,543

Examiner: A. Strange

Filed: 12/01/2000

Attorney Docket No.: 555255012129

For: Virtual Machine Web Browser

APPEAL BRIEF

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited today with the United States Postal Service as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on June 7, 2006.

By Debra Peyton.

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Appeal Brief is filed in response to the Final Office Action mailed September 15, 2005, which finally rejected pending claims 44-74 of this application. A Notice of Appeal and Request for Pre-Appeal Brief Conference was received by the U.S. Patent Office on January 12, 2006. A Notice of Panel Decision from Pre-Appeal Brief Review was mailed on March 8, 2006. A petition and fee for a two month time extension is included herewith.

I. Real Parties In Interest

The real party in interest is Research In Motion Limited as evidenced by an assignment recorded at Reel/Frame 011373/0948.

II. Related Appeals And Interferences

There are no related appeals or interferences to the instant application.

III. Status Of Claims

Claims 44-74 are pending and are finally rejected.

IV. Status Of Amendments

No amendments have been filed subsequent to the final rejection.

V. Summary Of Claimed Subject Matter

A. Independent Claim 44

Independent claim 44 describes a method of browsing content maintained in a page-rendered language without the use of a page-rendering browser application on a mobile communication device. The method includes the following steps:

> generating a request for content at the mobile communication device and transmitting the request to a gateway coupling the mobile communication device to a data network;
>
> forwarding the content request from the gateway to a server on the data network where the content is stored in the page-rendered language;

returning the requested page-rendered content from the server to the gateway;

translating the page-rendered content into a programmatic language and generating a corresponding executable program at the gateway; and

forwarding the executable program to the mobile communication device which executes the program in order to browse the requested content.

The method of claim 44 describes a technique for browsing content maintained in a page-rendered language, such as HTML, for example, without the use of a browser that is capable of rendering the page-rendered content. (Application, p. 4, ll. 18-20; p. 4, ll. 8-12; p. 5, ll. 15-22, p. 6, ll. 15-18, p. 7, ll. 4-6, p. 8, ll. 6-10) The method begins by generating a request for the page-rendered content at a mobile communication device and transmitting that request to a gateway system. (Application, p. 20, ll. 13-22; p. 3, ll. 13-16; p. 6, ll. 15-20; p. 12, ll. 11-14) The gateway system then forwards the request to a server on a data network where the content is stored in the page-rendered format. (Application, p. 5, ll. 15-20; p. 10, l. 12 through p. 11, l. 3; p. 13, ll. 9-16) The server returns the page-rendered content to the gateway, which then translates the page-rendered content into a programmatic language and generates an executable program therefrom. (Application, p. 13, ll. 17-22; p. 3, ll. 16-18; p. 4, ll. 3-7; p. 13, l. 4 through p. 17, l. 3) The executable program is then forwarded by the gateway to the mobile device, which executes the program to browse the request content. (Application, FIG. 3; p. 18, line 3 through p. 20, line 7; p. 21, l. 3 through p. 22, l. 14) In this manner, a mobile device without a page-rendering browser can browse content maintained on a data network in a page-rendered format.

B. Independent Claim 55

Claim 55 describes a system for browsing web pages maintained in a page-rendering language at a plurality of Internet web-servers. (Application, p. 4, ll. 18-20; p. 4, ll. 8-12; p. 5, ll. 15-22, p. 6, ll. 15-18, p. 7, ll. 4-6, p. 8, ll. 6-10) The system includes a wireless gateway system coupling a wireless data network to the Internet, the wireless gateway system including: (1) an interface component configured to receive web page requests from the wireless data network and to forward the web page requests to the plurality of Internet web-servers, the plurality of Internet web-servers responding to the web page requests and transmitting the requested web pages to the wireless gateway system; (Application, FIGs. 1, 2; p. 5, ll. 15-20; p. 10, l. 12 through p. 11, l. 3; p. 13, ll. 9-16; p. 20, ll. 13-22; p. 3, ll. 13-16; p. 6, ll. 15-20; p. 12, ll. 11-14) and (2) a translation component for translating the received web pages from the page-rendering language into a programmatic language that is capable of being executed. (Application, FIG. 2; p. 13, ll. 17-22; p. 3, ll. 16-18; p. 4, ll. 3-7; p. 13, l. 4 through p. 17, l. 3) The system also comprises a wireless mobile communication device configured to transmit web page requests to the wireless gateway system via the wireless data network and to receive translated web pages in the programmatic language in response thereto, the wireless mobile communication device including a program interpreter for executing programs formatted in the programmatic language. (Application, FIG. 3; p. 20, ll. 13-22; p. 3, ll. 13-16; p. 6, ll. 15-20; p. 12, ll. 11-14; p. 18, line 3 through p. 20, line 7; p. 21, l. 3 through p. 22, l. 14)

C. Independent Claim 64

Claim 64 describes a method of browsing web pages without the use of a web browser application capable of rendering web pages maintained in a page-rendered language format. (Application, p. 4, ll. 18-20; p. 4, ll. 8-12; p. 5, ll. 15-22, p. 6, ll. 15-18, p. 7, ll. 4-6, p. 8, ll. 6-10) The method includes the steps of: (A) transmitting a web page request to a web server that maintains web pages in the page-rendered language; (Application, p. 5, ll. 15-20; p. 10, l. 12 through p. 11, l. 3; p. 13, ll. 9-16) (B) forwarding the requested web page to a translation component; and (C) translating the requested web page from the page-rendered language format into a programmatic language format that is capable of being executed by a program interpreter; (Application, p. 13, ll. 17-22; p. 3, ll. 16-18; p. 4, ll. 3-7; p. 13, l. 4 through p. 17, l. 3) and (D) executing the translated web page in the programmatic language format using the program interpreter in order to browse the web page. (Application, FIG. 3; p. 18, line 3 through p. 20, line 7; p. 21, l. 3 through p. 22, l. 14)

D. Independent Claim 65

Independent claim 65 describes a mobile device for use with an information system that stores pages of information formatted in a page-rendering language, the information system including a plurality of information sources that store the pages of information formatted in the page-rendering language, and a translation component that translates the pages of information from the page rendering language format into a programmatic language format. (Application, p. 5, ll. 15-20; p. 10, l. 12 through p. 11, l. 3; p. 13, ll. 9-16; p. 13, ll. 17-22; p. 3, ll. 16-18; p. 4, ll. 3-7; p. 13, l. 4 through p. 17, l. 3) The mobile device incluldes: (1) a file explorer component for generating a request for an information page stored in the information system; (Application, FIGs. 3-5; p. 18, l. 21 throough p. 19, l. 8) and (2) a program interpreter for executing a program

returned by the information system in response to the generated request, wherein the program is generated by the translation component in the programmatic language format and represents the requested information page. (Application, FIG. 3; p. 18, line 3 through p. 20, line 7; p. 21, l. 3 through p. 22, l. 14)

E. Independent Claim 73

Claim 73 recites a method of browsing page-rendered documents on a mobile device without the use of a page-rendering browser application. (Application, p. 4, ll. 18-20; p. 4, ll. 8-12; p. 5, ll. 15-22, p. 6, ll. 15-18, p. 7, ll. 4-6, p. 8, ll. 6-10) The method includes the following steps: (A) generating a request for a page-rendered document at the mobile device; and (B) transmitting the request from the mobile device to a gateway system having a translation component; (Application, p. 20, ll. 13-22; p. 3, ll. 13-16; p. 6, ll. 15-20; p. 12, ll. 11-14) (C) forwarding the request from the gateway system to a server where the requested page-rendered document is stored; (Application, p. 5, ll. 15-20; p. 10, l. 12 through p. 11, l. 3; p. 13, ll. 9-16) (D) transmitting the requested page-rendered document from the server to the gateway system; and (E) translating the requested page-rendered document into a programmatic language that is capable of being executed; (Application, p. 13, ll. 17-22; p. 3, ll. 16-18; p. 4, ll. 3-7; p. 13, l. 4 through p. 17, l. 3) (F) forwarding the translated document in the programmatic language to the mobile device; and (G) executing the programmatic language of the translated document using a program interpreter at the mobile device in order to browse the page-rendered document without the use of a page-rendering browser. (Application, FIG. 3; p. 18, line 3 through p. 20, line 7; p. 21, l. 3 through p. 22, l. 14)

F. Independent Claim 74

Claim 74 recites a JAVA-based system for browsing HTML formatted content on the Internet. (Application, p. 15; p. 7, l. 18 through p. 8, l. 5; p. 8, ll. 14-24) The system includes: a gateway system for receiving requests for HTML formatted content and for fetching the HTML formatted content from a plurality of Internet servers, the gateway system having a translation component for converting HTML formatted content into JAVA programs; (Application, p. 5, ll. 15-20; p. 10, l. 12 through p. 11, l. 3; p. 13, ll. 9-16; p. 13, ll. 17-22; p. 3, ll. 16-18; p. 4, ll. 3-7; p. 13, l. 4 through p. 17, l. 3) and a mobile device having a file explorer component for transmitting a request for HTML formatted content to the gateway system and for receiving a JAVA program from the gateway system in return; and a JAVA virtual machine engine for executing the JAVA program in order to browse the HTML formatted content. (Application, FIG. 3; p. 20, ll. 13-22; p. 3, ll. 13-16; p. 6, ll. 15-20; p. 12, ll. 11-14; p. 18, line 3 through p. 20, line 7; p. 21, l. 3 through p. 22, l. 14)

VI. Grounds Of Rejection To Be Reviewed On Appeal

- Whether claims 44-46, 48-52, 55-57, 58-61, 64-70, 73 and 74 are unpatentable under 35 U.S.C. § 102(e) as being anticipated by Schwartz et al. (US 6,473,609).
- Whether claims 44-52, 55-57, 58-61, 64-70, 73 and 74 are unpatentable under 35 U.S.C. § 102(e) as being anticipated by Lowery (US 6,446,111).

VII. Argument

A. Independent claims 44, 55, 64, 65, 73 and 74 are Not Anticipated by Schwartz or Lowery

35 U.S.C. § 102(e) provides that a person shall be entitled to a patent unless the invention was described in a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent.

In order to establish a prima facie case of anticipation under 35 U.S.C. § 102(e), or any other provision of section 102, the prior art reference must teach every limitation of the claimed invention. *See*, MPEP 706.02; 2131. "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegall Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the . . . claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 2 USPQ2d 1913, 1920 (Fed. Cir. 1989).

The fundamental precept of an anticipation rejection is that every single claim limitation is found in a single prior art reference. *See, e.g., Structural Rubber Prod. Co. v. Park Rubber Co.*, 749 F.2d 707, 223 USPQ 1264 (Fed. Cir. 1984) (finding that anticipation can only be established by a single prior art reference which discloses each and every element of the claimed invention); and *In re Robertson*, 169 F.3d 743, 49 USPQ2d 1949 (Fed. Cir. 1999) ("Anticipation under 35 USC 102(e) requires that 'each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference.' (citation omitted)"). If even a single claim element is missing from the teaching of the prior art reference, then the anticipation rejection cannot stand. *Kloster Speedsteel AB v. Crucible Inc.*, 793 F.2d 1565, 230

USPQ 81 (Fed. Cir. 1986), *on rehearing*, 231 USPQ 160 (Fed. Cir. 1986), *cert. denied*, 479 U.S. 1034 (1987) (finding that the absence from a cited reference of any element of a claim of a patent negates anticipation of that claim by the reference.)

1. Schwartz Does Not Anticipate the Independent Claims

Schwartz describes a client-server system including a link server interposed between a wireless network and a wired network that operates like a proxy for a wireless client. The wireless client transmits an URL request to the proxy link server, which retrieves page-rendered code from a web server. This page-rendered code is then compacted into a smaller version of the page-rendered code using a converter. The compacted page-rendered code is then transmitted to a special interface engine on the client that is capable of rendering a display using the compacted data. Importantly, the interface engine in Schwartz is clearly a rendering engine and the compacted code generated by the proxy – which is referred to as SDD data in Schwartz – is not programmatic code that can be executed but is instead a series of page-rendering commands that are rendered by the interface engine:

> Schwartz, Abstract – "The interface engine typically performs tasks that do not require considerable computing power and memory, such as receiving input data from users, and the **rendering** of the compact data format received from the link server device, . . ."
>
> Schwartz, col. 3, ll. 55-60 – "in a method of the present invention, the link server sends over the wireless network a compact data file it generates, and **the interface engine renders the compact data file** to cause the display screen to display contents represented by the compact data file."
>
> Schwartz, col. 9, line 60 through col. 10, line 8 – "According to one embodiment, the SDD file is a group if Imp data. . . There are a set of rules or grammars in the Imp data that an interface engine, **upon rendering of the Imp data**, causes a screen to display the contents of the corresponding markup language file."

As these quotations from Schwartz demonstrate, the reference clearly does not disclose or suggest the possibility of converting page-rendered code into programmatic code that can be directly executed by the client device. Rather, in Schwartz, the page-rendered code is merely converted into a more compact form of smaller or less page-rendering code segments. The SDD data that is generated by the link server in the Schwartz reference is still page-rendered code, it is not programmatic code. As stated in Schwartz, "[t]ypically, the screen commands are expressed in a form of screen description data (SDD) **that is rendered** in an interface engine in mobile device 350." (Col. 9, ll. 36-40; **emphasis** supplied)

Furthermore, column 9, lines 50-58 of Schwartz provide an example "ASCII-like" representation of an SDD data file that clearly shows that it is a data file formatted in a page-rendered language, it is not an executable program. Many other portions of Schwartz support the fact that the SDD data file provided to the client device is just that – a data file – it is not an executable program: "Upon receiving and rendering **the screen description data**, the interface engine 616 causes display screen 618 to display the information embedded in the screen description data;" (Col. 11, ll. 44-47) ". . . while the interface engine in the terminal is only responsible **for rendering the screen description data** to cause the display screen to display contents and receive inputs from a user." (Col. 12, ll. 52-56)

Finally, the final office action states that "Schwartz clearly discloses that the SDD data may be directly rendered by the interface engine of the client device, 'without further processing.'" This is an admission that Schwartz does not, in fact, translate page-rendered code into executable programmatic code, but rather converts page-rendered code into other page-rendered code that must be **rendered**. This is exactly the point made above, and clearly demonstrates why the anticipation rejection over Schwartz is in error.

The independent claims recited above all require that the page-rendered code is translated into programmatic code that can be executed at the mobile device, without the use of a page-rendering application. Schwartz clearly requires a page-rendereing application to "directly render" its SDD data file, and therefore does not meet all the limitations of these claims. The anticipation rejection, therefore, should be reversed.

2. Lowery Does Not Anticipate the Independent Claims

Lowery is directed to the problem of transmitting dynamic web pages – which may include video or animations – over a network to a small portable device, such as a PDA having limited memory and processing capabilities. (Lowery, col. 5, ll. 1-60) Due to the limited capabilities of the PDA, Lowery teaches that it is difficult to transmit such dynamic web pages to the PDA because it typically cannot operate the browser plug-ins that are required to interpret the video or animation data. (Lowery, col., 5, ll. 40-60) Lowery proposes to alleviate this problem by using JAVA applets in place of the browser plug-ins. (Lowery, col. 60, line 1 through col. 6, line 27; Col. 7, ll. 1-5; claim 1) The use of applets, however, may cause a delay problem according to Lowery because of the need for the applet to retrieve additional data from a server to properly operate:

> "However, Java applets, once transferred, typically must go back out over the wireless network two or more times. Once, to get classes and objects needed by the applet, and again to get the very data that the Java applet requires to make the Java applet useful to the user of the PDA. Thus, not only has the Java applet user had to wait for the Java applet to transfer to the PDA, but must wait a further period of time for the Java applet to transfer needed classes, objects and data over the network to the PDA." (Lowery, col. 7, line 62 through col. 8, line 4)

In order to cure this "delay" problem associated with generating an applet instead of a dynamic web page, Lowery proposes to encapsulate all of the required data – i.e., the mentioned classes and objects required by the applet – into the applet itself so that the server can respond to

the client request with a single transmission of the applet instead of the dynamic web page: "According to the teachings of the present invention, a Java applet can be augmented to address this problem by including all or most of the required data and the associated functionality in a single transmission to the client." (Lowery, col. 8, ll. 5-9)

As described above, Lowery describes a method of modifying a specific web server so that it generates an applet having encapsulated data objects instead of generating a dynamic web page requiring browser plug-ins. Figure 1 of Lowery shows such a configuration in which the client 12 is in direct communication with the modified web server 18. The modified web server 18 can retrieve data from other sources 22, 24, and may use this data in the course of generating the executable applet that is provided to the client.

The applet is constructed at the specific web server, and may include data items from other sources that are retrieved over a network. Although Lowery's web server may obtain this "data" from other sources, the reference does not disclose or suggest that this "data" is content maintained in a page-rendered language, nor does it disclose the translation of the page-rendered content into a programmatic language. Instead, Lowery indicates that this "data" is the data required to operated the applet – i.e., the classes and objects required to run the applet.

Lowery does not disclose or suggest the translation of standard page-rendering language documents, such as HTML documents, into a programmatic language, such as JAVA or some other form of executable program code. Moreover, there is no translation at all in Lowery, because Lowery generates executable code directly, it does not translate fetched HTML or any other type of page-rendered code into a programmatic code. The system described in Lowery could not be utilized with any web site **other than** one that has been specifically modified to generate the executable applet in lieu of the dynamic web page, whereas the system and method

described and claimed in the present application could be used with virtually any web site due to the provision of the translation function which converts the retrieved page-rendered code into executable program code.

As described in the independent claims, the plurality of web sites or other information sources to be accessed maintain web page data in standard HTML or other types of standard page-rendering formats, and therefore these web sited do not require any special modifications. The gateway/proxy computer, or other part of the system, retrieves the standard HTML-type web pages or other information pages from these plurality of sources, and then translates the page-rendering content into a programmatic language for execution by the client mobile device. In this manner, the client machine can access virtually any web site, regardless of its formatting, because the process of translating the page-rendering code into programmatic code has been centralized at the gateway or other part of the system. Lowery's system, by distinction, can only access content that is generated by a modified web server that is able to generate an executable applet according to Lowery's specifications.

The final office action points to the following sections of Lowery in support of the assertion that the reference teaches an intermediary system, such as a gateway, that translates page-rendered content retrieved from a server into programmatic code:

> The basic concept of the web page is that there exists a division of responsibility between the client's web browser and the server's web page. The web browser typically locates, retrieves and displays the web pages, executes hyperlinks and applets, and generally interprets web page information. The web page comprises the raw data, hyperlinks, and HTML constructs that may be executed by the web browser. (Col. 7, lines 17-24)
>
> The applet 26, in the disclosed embodiment, may comprise a Java, ActiveX or other suitable type of applet which can be executed by the client. (Col. 9, lines 34-36)

> The method proceeds to step 52 where the server 18 collects the data items 28 that represent an appropriate response to the request from the client 12. The web server application 20 may collect the data from the plurality of data sources 22 through 24 and server 18. The method proceeds to step 53 where the web server application 20 or an external application generates the applet 26, once the appropriate data items 28 have been collected. (Col. 15, lines 24-32)

Clearly missing from these portions of Lowery is any mention of the server 18 retrieving HTML or other page-rendered content from the "data sources" 22 through 24. Rather, the reference merely states that certain undefined and non-described "data" is collected, but does not specify what type of data this may be. As noted above, however, other portions of Lowery teach that this data includes the class and object data necessary to run the applet, which is not the same as retreiving page-rendered content from a plurality of different web servers. Lowery clearly does not state that the "data" retrieved from the "data sources" is page-rendered content that is subsequently translated from the page-rendered format into programmatic code. Thus, the reference does not anticipate any of the independent claims and the rejection should be reversed.

VIII. Claims Appendix

A claims appendix containing a copy of the claims subject to this appeal is attached.

IX. Evidence Appendix

No evidence is being submitted pursuant to 37 C.F.R. §§ 1.130, 1.131, or 1.132, nor is there any other evidence entered by the Examiner or relied upon by the Applicant. An evidence appendix indicating "None" is attached.

X. Related Proceedings Appendix

There are no related proceedings. An related proceedings appendix indicating "None" is attached.

Respectfully submitted,

Date: 6/7/2006

By: David B. Cochran

David B. Cochran, Reg. No. 39,142
Jones Day
North Point
901 Lakeside Ave.
Cleveland, Ohio 44114
(216) 586-1162

CLAIMS APPENDIX

Claims 1-43 (Cancelled)

44. (Previously Presented) A method of browsing content maintained in a page-rendered language without the use of a page-rendering browser application on a mobile communication device, comprising:

generating a request for content at the mobile communication device and transmitting the request to a gateway coupling the mobile communication device to a data network;

forwarding the content request from the gateway to a server on the data network where the content is stored in the page-rendered language;

returning the requested page-rendered content from the server to the gateway;

translating the page-rendered content into a programmatic language and generating a corresponding executable program at the gateway; and

forwarding the executable program to the mobile communication device which executes the program in order to browse the requested content.

45. (Previously Presented) The method of claim 44, further comprising:

compressing the executable program prior to forwarding it to the mobile communication device.

46. (Previously Presented) The method of claim 45, further comprising:

providing a byte code generator at the gateway; and

executing the byte code generator to compress the translated programmatic language into byte codes.

47. (Previously Presented) The method of claim 44, wherein the request for content is a Uniform Resource Locator request which is transmitted to the gateway using the Hypertext Transfer Protocol.

48. (Previously Presented) The method of claim 44, wherein the page-rendered language is HTML, HDML, XML or WML.

49. (Previously Presented) The method of claim 44, wherein the page-rendered content returned from the server to the gateway is one of a plurality of content types, the method further comprising:

providing a plurality of content translators at the gateway, each of the plurality of content translators translating page-rendered content of a particular content type into a common type of programmatic language;

determining the content type of the page-rendered content returned from the server to the gateway; and

selecting and executing one of the plurality of content translators at the gateway in correspondence with the determined type of content in order to translate the page-rendered content into the programmatic language.

50. (Previously Presented) The method of claim 44, wherein the mobile communication device is coupled to the gateway via a wireless data network.

51. (Previously Presented) The method of claim 44, wherein the programmatic language is a virtual machine language, the method further comprising:

providing a virtual machine at the mobile communication device for executing programs coded in the virtual machine language.

52. (Previously Presented) The method of claim 44, further comprising:

providing a file explorer at the mobile communication device, the file explorer generating the request for content and storing the executable program forwarded from the gateway.

53. (Previously Presented) The method of claim 52, further comprising:

generating a second request for content at the mobile communication device using the file explorer;

determining if an executable program representing the requested content is stored at the mobile communication device;

if the executable program is stored at the mobile communication device, then executing the program in order to browse the requested content;

if the executable program is not stored at the mobile communication device, then transmitting the second request for content to the gateway.

54. (Previously Presented) The method of claim 44, further comprising:

verifying the executable program at the mobile communication device prior to executing the program.

55. (Previously Presented) A system for browsing web pages maintained in a page-rendering language at a plurality of Internet web-servers, comprising:

a wireless gateway system coupling a wireless data network to the Internet, the wireless gateway system including:

an interface component configured to receive web page requests from the wireless data network and to forward the web page requests to the plurality of Internet web-servers, the plurality of Internet web-servers responding to the web page requests and transmitting the requested web pages to the wireless gateway system; and

a translation component for translating the received web pages from the page-rendering language into a programmatic language that is capable of being executed; and

a wireless mobile communication device configured to transmit web page requests to the wireless gateway system via the wireless data network and to receive translated web pages in the programmatic language in response thereto, the wireless mobile communication device including a program interpreter for executing programs formatted in the programmatic language.

56. (Previously Presented) The system of claim 55, wherein the wireless gateway system further comprises a data compressor for compressing the translated web pages prior to transmitting them to the wireless mobile communication device.

57. (Previously Presented) The system of claim 56, wherein the data compressor includes a byte code generator for converting the translated web pages into a compressed byte code format.

58. (Previously Presented) The system of claim 55, wherein the page-rendered language is HTML, HDML, XML or WML.

59. (Previously Presented) The system of claim 55, wherein the page-rendered web pages received from the Internet web-servers are formatted using one of a plurality of content types, the system further comprising:

a plurality of translation components at the wireless gateway system, each of the plurality of translation components configured to translate page-rendered content of a particular content type into a common type of programmatic language;

wherein the wireless gateway system determines the content type of the page-rendered web pages received from the Internet web-servers and executes one of the plurality of translation components in correspondence with the determined content type in order to translate the web-page into the programmatic language.

60. (Previously Presented) The system of claim 55, wherein the programmatic language is a virtual machine language, the system further comprising a virtual machine operating at the wireless mobile communication device for executing programs coded in the virtual machine language.

61. (Previously Presented) The system of claim 55, further comprising a file explorer component operating at the wireless mobile communication device for generating the web page requests for web pages and for storing the translated web pages received from the wireless gateway system.

62. (Previously Presented) The system of claim 61, wherein the file explorer component generates a second web page request and determines if a translated web page corresponding to the requested web page is stored at the wireless mobile communication device, and if so, then the program interpreter executes the stored translated web page.

63. (Previously Presented) The system of claim 55, wherein the wireless mobile communication device verifies the programs formatted in the programmatic language prior to execution.

64. (Previously Presented) A method of browsing web pages without the use of a web browser application capable of rendering web pages maintained in a page-rendered language format, comprising:

transmitting a web page request to a web server that maintains web pages in the page-rendered language;

forwarding the requested web page to a translation component;

translating the requested web page from the page-rendered language format into a programmatic language format that is capable of being executed by a program interpreter; and

executing the translated web page in the programmatic language format using the program interpreter in order to browse the web page.

65. (Previously Presented) A mobile device for use with an information system that stores pages of information formatted in a page-rendering language, the information system including a plurality of information sources that store the pages of information formatted in the page-rendering language, and a translation component that translates the pages of information from the page rendering language format into a programmatic language format, the mobile device comprising:

a file explorer component for generating a request for an information page stored in the information system; and

a program interpreter for executing a program returned by the information system in response to the generated request, wherein the program is generated by the translation component in the programmatic language format and represents the requested information page.

66. (Previously Presented) The mobile device of claim 65, wherein the information page is a web page formatted in HTML, HDML, WML or XML.

67. (Previously Presented) The mobile device of claim 65, wherein the program returned by the information system is compressed, the mobile device comprising a decompression component for decompressing the compressed program.

68. (Previously Presented) The mobile device of claim 67, wherein the program is compressed using a byte code generator.

69. (Previously Presented) The mobile device of claim 65, wherein the program interpreter is a

virtual machine capable of executed programs coded in a virtual machine language.

70. (Previously Presented) The mobile device of claim 69, wherein the virtual machine is a JAVA virtual machine.

71. (Previously Presented) The mobile device of claim 65, wherein the file explorer component is operable to:

generate a second request for an information page;

determine if an executable program representing the requested information page is stored at the mobile device;

execute the stored program in order to browse the information page if the executable program is stored at the mobile device; and

transmitting the second request for the information page to the information system if the executable program is not stored at the mobile device.

72. (Previously Presented) The mobile device of claim 65, further comprising a verification component for verifying the executable program prior to execution.

73. (Previously Presented) A method of browsing page-rendered documents on a mobile device without the use of a page-rendering browser application, comprising:

generating a request for a page-rendered document at the mobile device;

transmitting the request from the mobile device to a gateway system having a translation component;

forwarding the request from the gateway system to a server where the requested page-rendered document is stored;

transmitting the requested page-rendered document from the server to the gateway system;

translating the requested page-rendered document into a programmatic language that is capable of being executed;

forwarding the translated document in the programmatic language to the mobile device; and

executing the programmatic language of the translated document using a program interpreter at the mobile device in order to browse the page-rendered document without the use of a page-rendering browser.

74. (Previously Presented) A JAVA-based system for browsing HTML formatted content on the Internet, comprising:

a gateway system for receiving requests for HTML formatted content and for fetching the HTML formatted content from a plurality of Internet servers, the gateway system having a translation component for converting HTML formatted content into JAVA programs; and

a mobile device having a file explorer component for transmitting a request for HTML formatted content to the gateway system and for receiving a JAVA program from the gateway system in return; and a JAVA virtual machine engine for executing the JAVA program in order to browse the HTML formatted content.

EVIDENCE APPENDIX

NONE

(No evidence is being submitted pursuant to 37 C.F.R. §§ 1.130, 1.131, or 1.132, nor is there any other evidence entered by the Examiner or relied upon by the Applicant)

RELATED PROCEEDINGS APPENDIX

NONE

(There are no related proceedings)